

FIG. 1

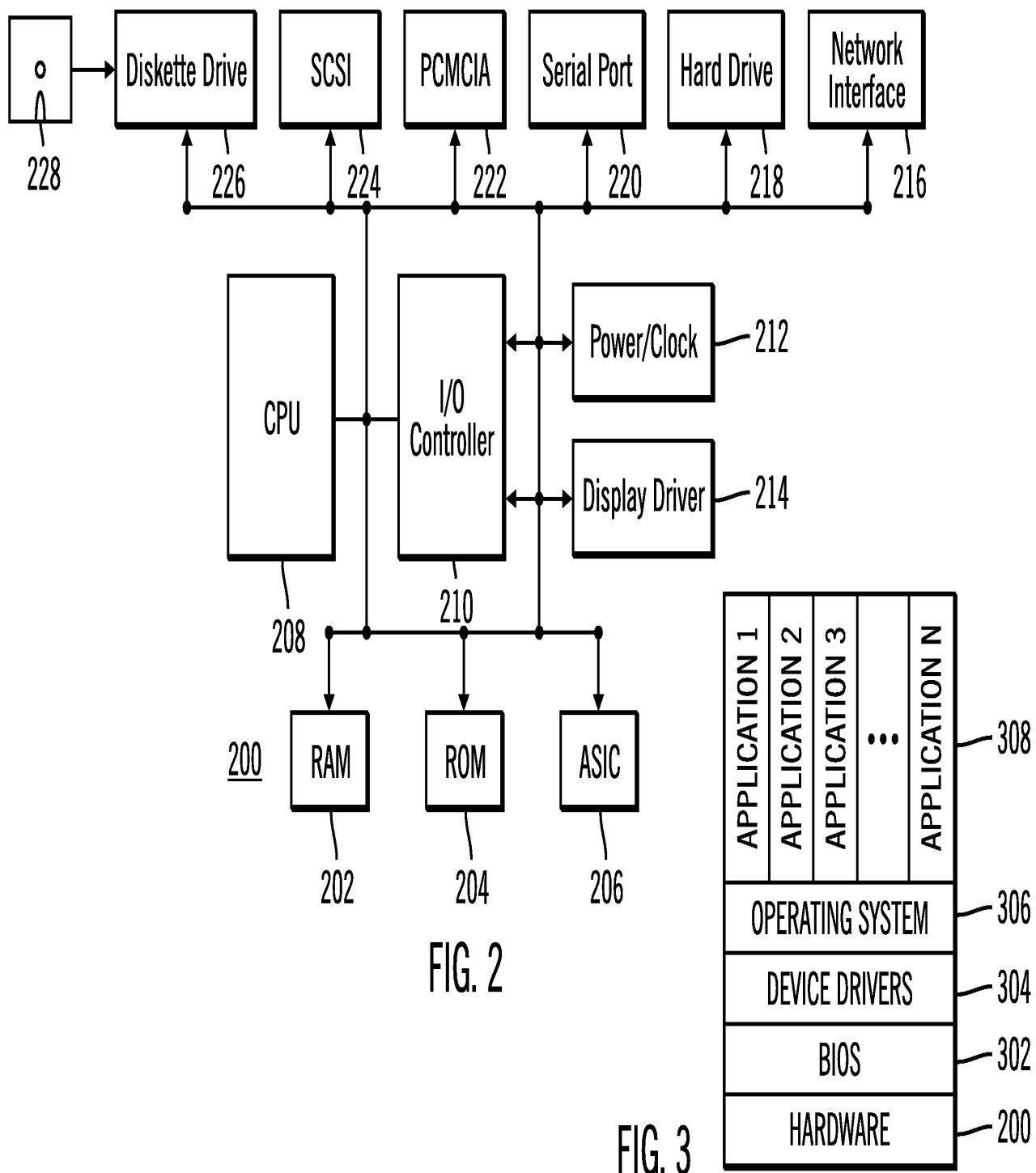
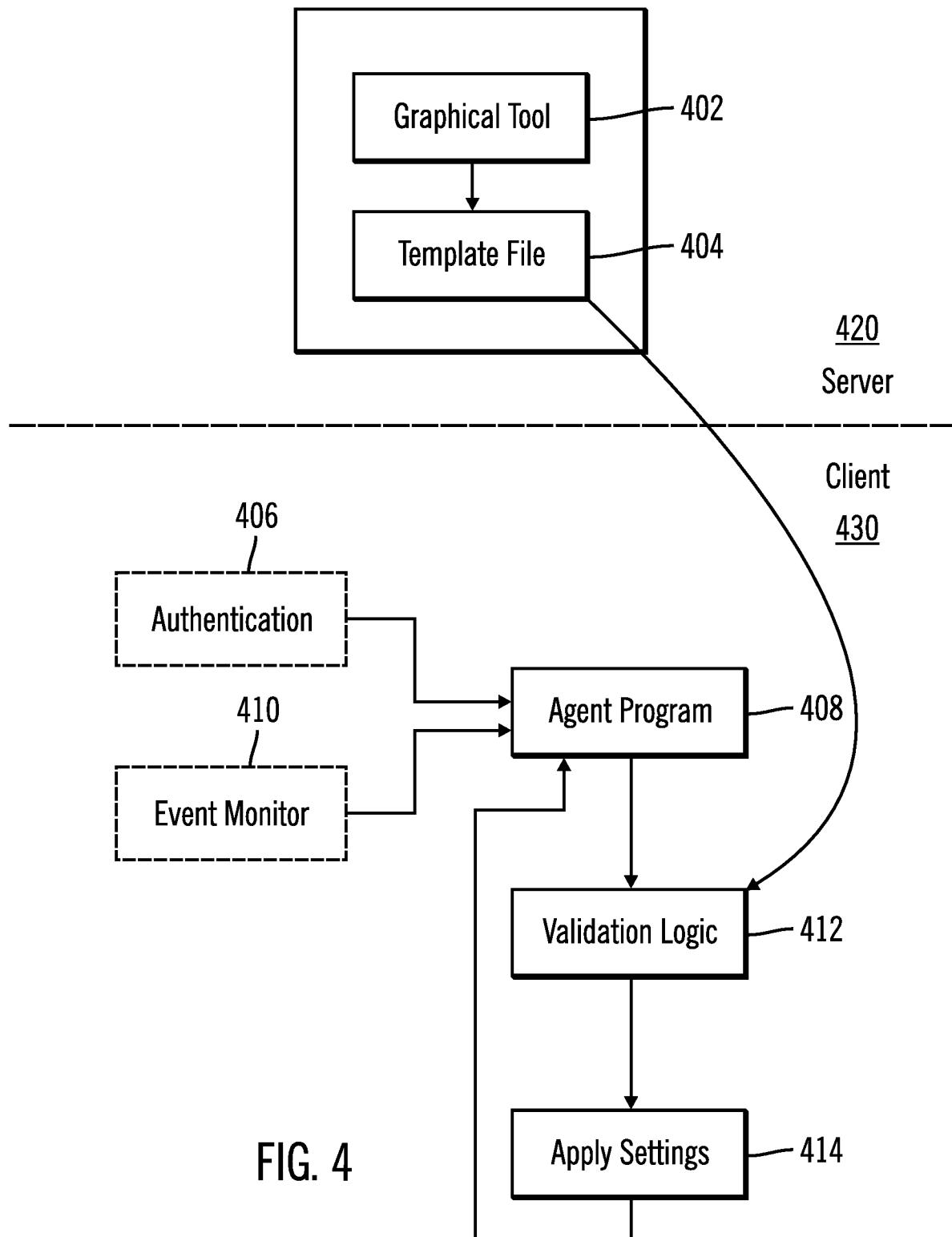
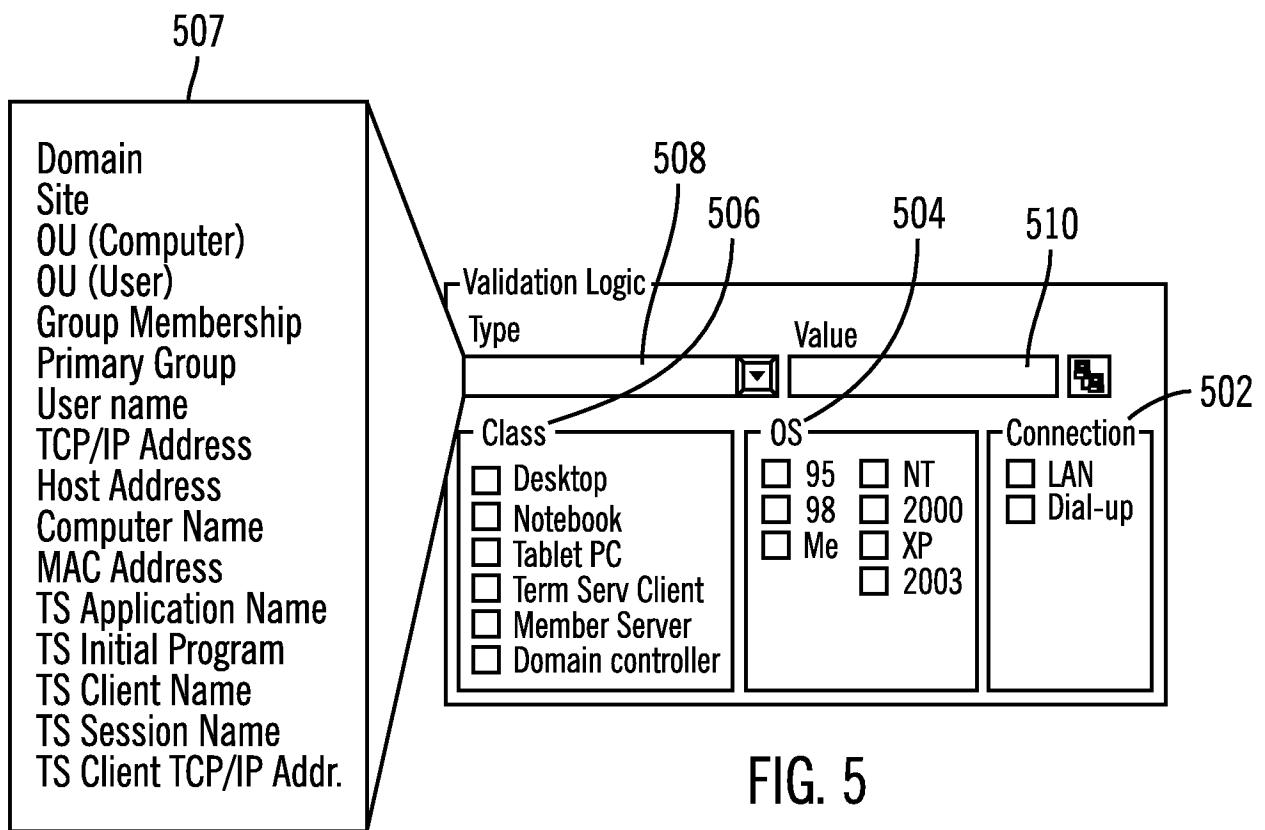


FIG. 2

FIG. 3





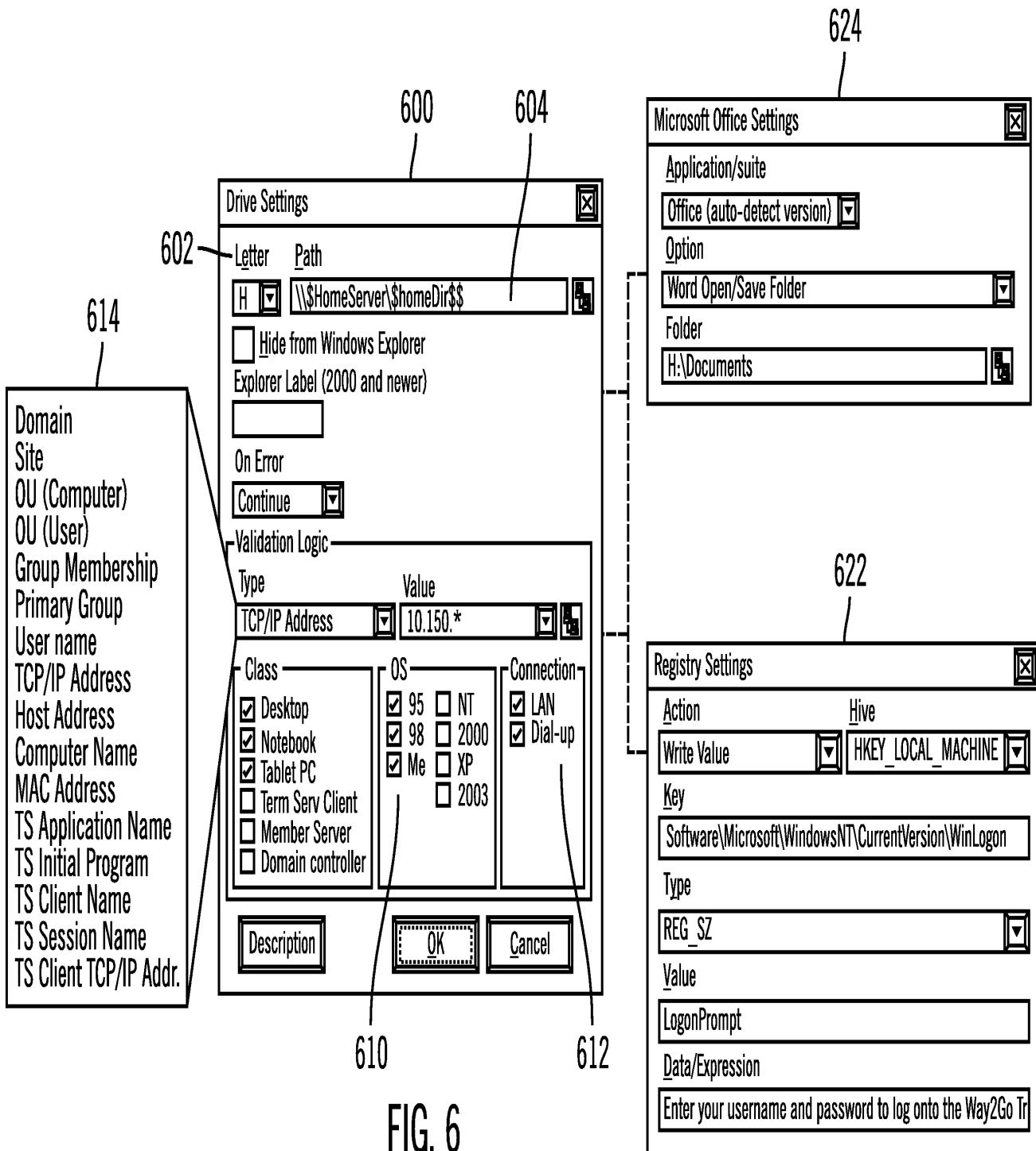


FIG. 6

View Pane

Application Launcher

Description	Filespec	Args	Cycle	Cycle Data	Frequency	Timing	Hide	Wait	Admin	Validation
testapp	arg	E	*	E	After	Visible	Continue	User	/G=!Accounting	

Settings Validation Logic

Validation

Type	<input type="checkbox"/> NOT	Value	
Group Membership	<input type="button" value="▼"/>	<input type="button" value="▼"/> <input type="button" value="□"/>	
<input type="button" value="Add"/>	<input type="button" value="Remove"/>	<input type="radio"/> OR <input type="radio"/> AND	
Operator	Type	Validation	
IF	Group	!Accounting Group	
AND NOT	Primary Group	!Human Resources Group	
OR	UserOU	RD-*	

Class

<input checked="" type="checkbox"/> Desktop
<input checked="" type="checkbox"/> Notebook
<input checked="" type="checkbox"/> Tablet PC
<input type="checkbox"/> Term Serv Client
<input checked="" type="checkbox"/> Member Server
<input checked="" type="checkbox"/> Domain controller

OS

<input checked="" type="checkbox"/> 95	<input type="checkbox"/> NT
<input checked="" type="checkbox"/> 98	<input checked="" type="checkbox"/> 2000
<input checked="" type="checkbox"/> Me	<input checked="" type="checkbox"/> XP

Connection

<input checked="" type="checkbox"/> LAN	<input type="checkbox"/> Dial-up
---	----------------------------------

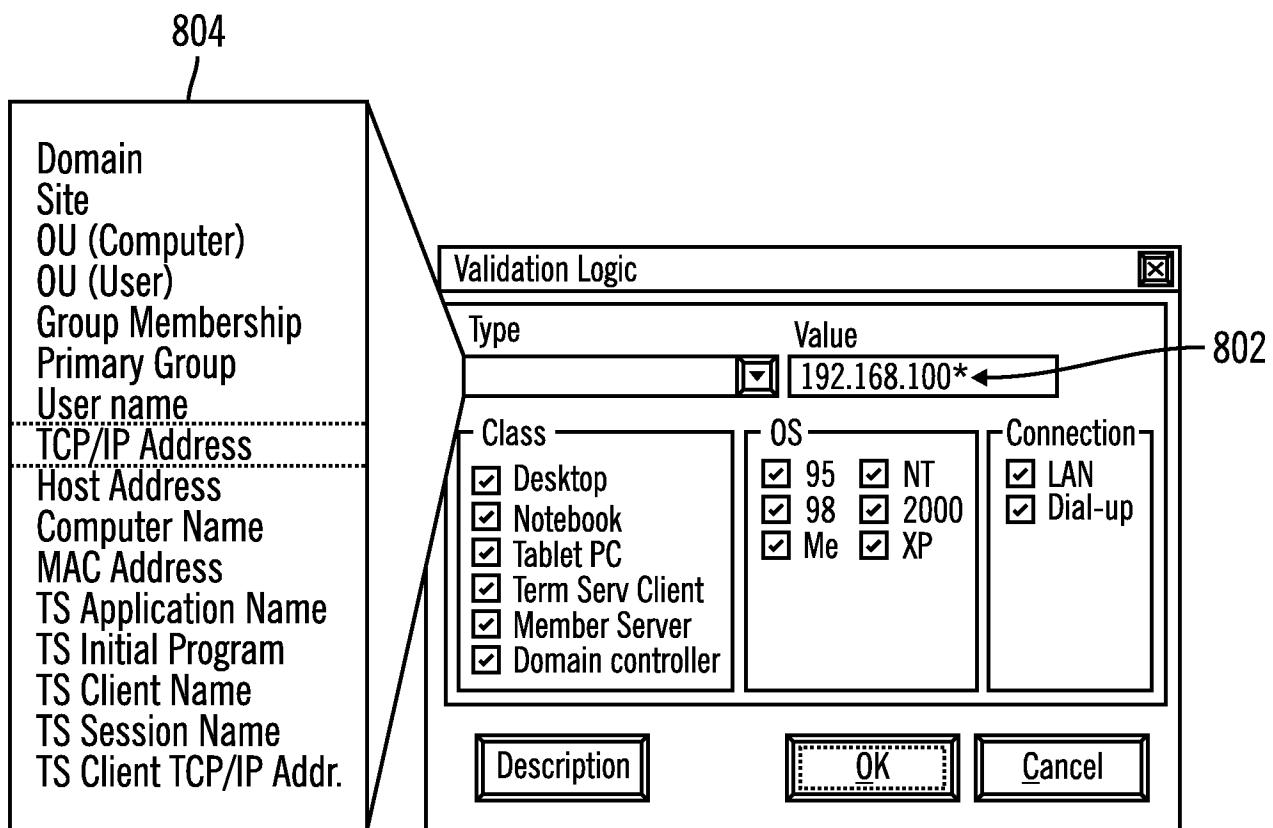


FIG. 8



8/13

```
function slMultiCompare($StringA,$StringB)
; SL platforms: 4.01 ; LastRev: 2002-Aug-21
; dependencies: slWildCompare(), slQuestionCompare()
; compares one string to another, and supports '*' and '?' as a wildcards
; stringA: constant string
; stringB: variable string
;           stringB can contain wildcards '*' and '?'
;           stringB can be an array or a single string containing multiple elements,
each separated by a semi-colon
dim $ArrayB, $elementB
$slMultiCompare=0 ; default false
if $StringA and $StringB
$StringA=trim($StringA)
if vartype($StringB)<8192 ; StringB is a string
$ArrayB=split($StringB+','','') ; remove last ; added for split to achieve at least
one element
    redim preserve $ArrayB[ubound($ArrayB)-1]
else ; StringB is an array
$ArrayB=$StringB
endif
for each $ElementB in $ArrayB
$ElementB=trim($elementB)
select
    case $ElementB='*' ; single wildcard - matches everything
        $slMultiCompare=1
        return ; true
    case $StringA=$ElementB
        $slMultiCompare=1
        return ; true
    case instr($ElementB,'*')
        if slWildCompare($StringA,$ElementB)
            $slMultiCompare=1
            return ; true
        endif
    case instr($ElementB,'?')
        if slWildCompare($StringA,$ElementB)
            $slMultiCompare=1
            return ; true
        endif
    case 1 ; no wildcards and we've already determined that strings don't match
        ; do nothing - proceed to next array element
    endselect
next
endif
endfunction

function slWildCompare($StringA,$StringB)
; SL platforms: 4.01 ; LastRev: 2002-Aug-21
; dependencies: slQuestionCompare()
; Do not call this function directly -- use slMultiCompare() instead
; compares one string to another, and supports wildcards
; stringA: constant string
; stringB: variable string (can contain wildcards '*' and '?')
; could add case-sensitivity option in future...
dim $LenStringA, $lenStringB, $QuestionLoc, $AsteriskLoc
dim $GlobArray, $LenGAE, $lenGAEfirst, $lenGAElast, $GAUB
$slWildCompare=0 ; default to no match
if $StringA and $StringB
$StringA=trim($StringA)
$LenStringA=len($StringA)
if $StringB='*' ; single wildcard - matches everything
```

FIG. 9



A/B

A/B

```

$slWildCompare=1
    return ;true
endif
if $StringA==$StringB ; exact match
    $slWildCompare=1
    return ;true
else ; not exact match
    $asteriskLoc=instr($StringB,'*')
    $questionLoc=instr($StringB,'?')
    if not ($asteriskLoc or $questionLoc)
        return ; false: no wildcards - no reason to continue
    endif
    $lenStringB=len($StringB)
    $GlobArray=split($StringB+'*','*')
    $GAUB=ubound($GlobArray)-1
    redim preserve $GlobArray[$GAUB] ; remove last * added for split to achieve at
least one element
    ; first Glob - special case test
    $lenGAEfirst=len($GlobArray[0])
    if not slQuestionCompare(left($StringA,$lenGAEfirst),$GlobArray[0])
        return ; false
    endif
    ; last Glob - special case test
    $lenGAElast=len($GlobArray[$GAUB])
    if not slQuestionCompare(right($StringA,$lenGAElast),$GlobArray[$GAUB])
        return ; false
    endif
    $StringA=substr($StringA,$lenGAEfirst+1,len($StringA)-$lenGAElast) ; removed final
-1 (was failing on *abc*)
    if $GAUB<2 ; less than 2 Globs - preceeding special case tests determined result
        $slWildCompare=1
        return ; true
    endif
    for $index=1 to $GAUB-1 ; process elements 2 through next-to-last
        $lenGAE=len($GlobArray[$index])
        if len($StringA)<$lenGAE
            return ; false
        endif
        while len($StringA) and not
            slQuestionCompare(left($StringA,$lenGAE),$GlobArray[$index])
            $StringA=substr($StringA,2)
        loop
        if not slQuestionCompare(left($StringA,$lenGAE),$GlobArray[$index])
            return ; false
        else
            $StringA=substr($StringA,$lenGAE+1)
        endif
    next
    $slWildCompare=1
endif
endif
endif
endfunction

function slQuestionCompare($StringA,$StringB)
; SL platforms: 4.01 ; LastRev: 2002-Aug-21
; Do not call this function directly -- use slMultiCompare() or slWildCompare() instead
; compares one string to another, and supports '?' as a wildcard
; StringA - constant
; StringB - variable
dim $index, $StringBchar
$slQuestionCompare=1
if $StringA and $StringB
    if $StringA==$StringB
        $slQuestionCompare=1 ; true

```

FIG. 9

B/C

B/C



```
else
    $slQuestionCompare=0 ; default no match
    if not instr($StringB,'?') ; no question marks
        return ; false
    else
        ; length of both strings must be same to continue
        if len($StringA)<>len($StringB) ; different lengths
            return ; false
        endif
        ; perform comparison character-by-character
        for $index=1 to len($StringA)
            $StringBchar=substr($StringB,$index,1)
            if (substr($StringA,$index,1)<>$StringBchar) and $StringBchar<>'?'
                return ; false
            endif
        next
        $slQuestionCompare=1 ; true
    endif
endif
endif
endfunction
```

FIG. 9

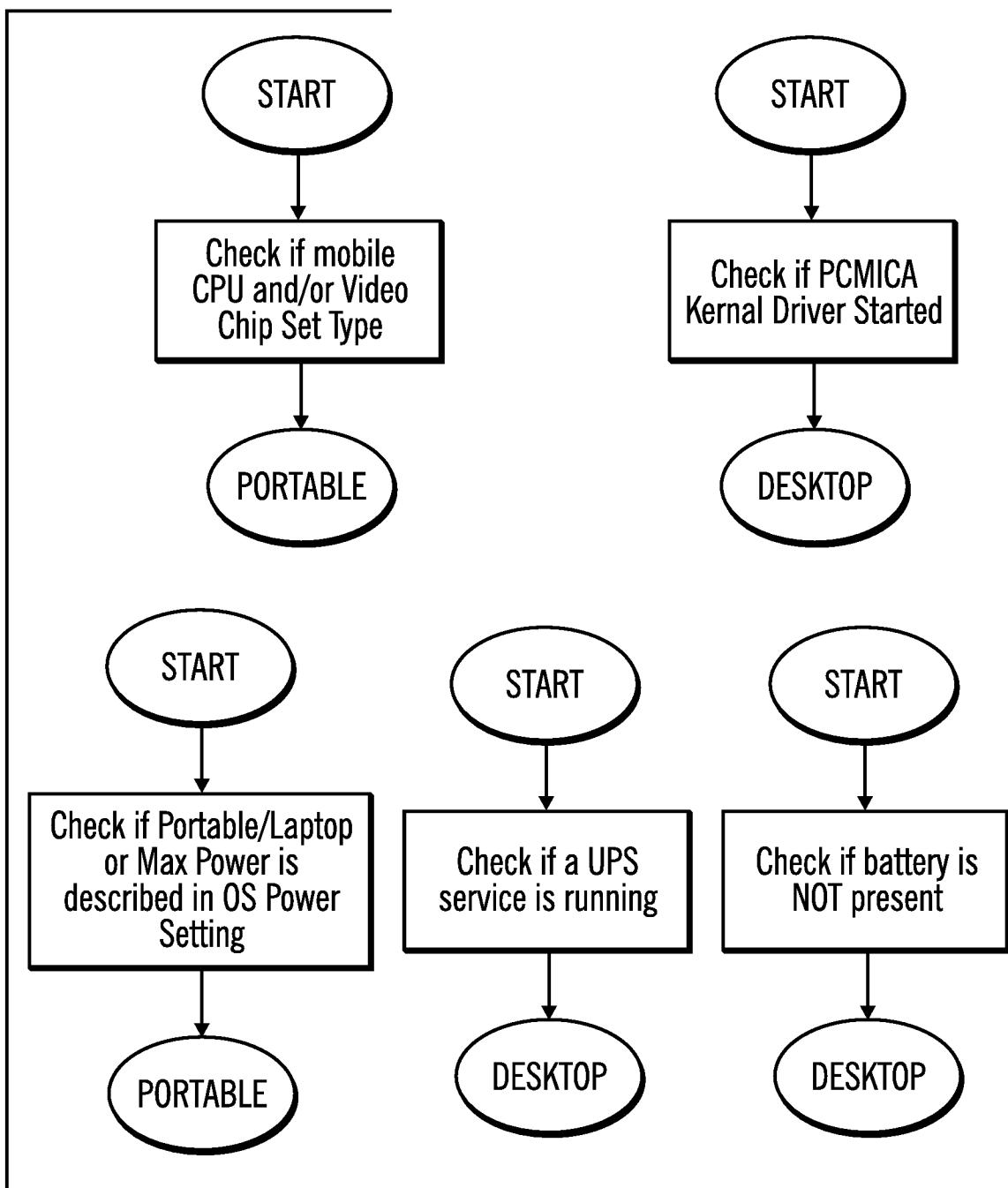


FIG. 10

```

$CurrentPowerProfileValue=readvalue('HKCU\Control
Panel\PowerCfg','CurrentPowerPolicy')
$CurrentPowerProfileName=readvalue('HKCU\Control
Panel\PowerCfg\PowerPolicies\'+$CurrentPowerProfileValue,'Name')
select
  case instr($SiProcessorNameString,'mobile') ; Mobile CPU type
    ; highly confident that this is a portable computer!
    ; platforms tested on: XP
    $ClientClassRule='rule 1: Mobile CPU type -> portable'
    $SiComputerType='Portable'
    $ClientClass='Port'
  case @INWIN=1 and
0+readvalue('HKLM\System\CurrentControlSet\Services\pcmcia','Start')=4 ; NT & PCMCIA
kernel driver not started
    ; highly confident that this is a desktop computer!
    ; platforms tested on: NT, 2000, XP
    $ClientClassRule='rule 2: PCMCIA driver not started (NT) -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case @INWIN=2 and
'+readvalue('HKLM\System\CurrentControlSet\Control\InstalledFiles','PCCard.vxd')=' ; 9x
& PCMCIA kernel driver not started
    ; highly confident that this is a desktop computer!
    ; platforms tested on: 95, 98, Me
    $ClientClassRule='rule 3: PCMCIA driver not started (9x) -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case $OS<>'NT' and $SiBatteryState=128 ; no battery present
    ; fairly confident that this is a desktop computer (it could be a laptop with the
battery removed).
    ; platforms tested on:
    $ClientClassRule='rule 4: No system battery detected -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case slGetServiceStartup('UPS')='Automatic' ; Built-in UPS service on 2000/XP
    ; highly confident that this is a desktop computer (who'd install UPS software on
a laptop?)
    ; platforms tested on: XP, 2000
    $ClientClassRule='rule 5: built-in UPS service is automatic -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case slGetServiceStartup('LiebertM')='Automatic' ; Liebert MultiLink 3.0
    ; highly confident that this is a desktop computer (who'd install UPS software on
a laptop?)
    ; platforms tested on: XP, 2000
    $ClientClassRule='rule 6: Liebert MultiLink UPS service is automatic -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case slGetServiceStartup('APCPBEEAgent')='Automatic' ; APC PowerChute Business
Edition 6.1
    ; highly confident that this is a desktop computer (who'd install UPS software on a
laptop?)
    ; platforms tested on: XP, 2000
    $ClientClassRule='rule 7: APC PowerChute Business Edition UPS service is
automatic -> desktop'
    $SiComputerType='Desktop'
    $ClientClass='Desk'
  case slGetServiceStartup('APC UPS Service')='Automatic' ; APC PowerChute Personal
Edition
    ; highly confident that this is a desktop computer (who'd install UPS software on
a laptop?)

```

A/B

A/B

```

; platforms tested on: XP, 2000
$ClientClassRule='rule 8: APC PowerChute Business Edition UPS service is
automatic -> desktop'
$SiComputerType='Desktop'
$ClientClass='Desk'
case $CurrentPowerProfileName='APC USB UPS'
; highly confident that this is a desktop computer (who'd install UPS software on
a laptop?)
; ***$$ what about other UPS brands? What about APC non-USB models?
; platforms tested on: XP, 2000
$ClientClassRule='rule 9: APC USB UPS power scheme -> desktop'
$SiComputerType='Desktop'
$ClientClass='Desk'
case $CurrentPowerProfileName='Portable/Laptop' or $CurrentPowerProfileName='Max
Battery'
; somewhat confident that this is a portable computer. This setting is user
profile-specific and can be changed
; platforms tested on: XP, 2000
$ClientClassRule='rule 10: portable/laptop or max battery power scheme ->
portable'
$SiComputerType='Portable'
$ClientClass='Port'
case 1
; At this point, here is what we know:
; Not a mobile CPU type
; The Portable/Laptop power scheme is not selected
; It does have PCMCIA sockets.
; 9x, 2000 & XP systems do not have a battery present
;
$ClientClassRule='rule 11: default -> portable'
$SiComputerType='Portable'
$ClientClass='Port'
endselect

```

FIG. 11